IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Single Window Navigation Methods and Systems

Inventor(s):
Scott Ruthfield
Richard Wolf
Satoshi Nakajima

ATTORNEY'S DOCKET NO. MS1-560US

RELATED APPLICATIONS

The following patent applications are related to the present application, are assigned to the assignee of this patent application, and are expressly incorporated by reference herein:

- U.S. Patent Application Serial No. _____, entitled "Methods and Systems of Providing Information to Computer Users", bearing attorney docket number MS1-557us, and filed on the same date as this patent application;
- U.S. Patent Application Serial No. _____, entitled "Methods, Systems, Architectures and Data Structures For Delivering Software via a Network", bearing attorney docket number MS1-559us, and filed on the same date as this patent application;
- U.S. Patent Application Serial No. _____, entitled "Network-based Software Extensions", bearing attorney docket number MS1-563us, and filed on the same date as this patent application;
- U.S. Patent Application Serial No. _____, entitled "Authoring Arbitrary XML Documents using DHTML and XSLT", bearing attorney docket number MS1-583us, and filed on the same date as this patent application;
- U.S. Patent Application Serial No. _____, entitled "Architectures For And Methods Of Providing Network-based Software Extensions", bearing attorney docket number MS1-586us, and filed on the same date as this patent application.
- U.S. Patent Application Serial No. _____, entitled "Task Sensitive Methods And Systems For Displaying Command Sets", bearing attorney docket number MS1-562us, and filed on the same date as this patent application.

TECHNICAL FIELD

This invention relates generally to computing systems and methods, and more particularly concerns systems and methods that facilitate creation and use of information within a computing environment.

BACKGROUND

The current popular paradigm in software application design is to provide a separate and oftentimes different window for each application that might be executing on a computing device. When a user desires to use a particular application, they will typically open up the application which will then be presented by the computer system in the form of a window or windows that allows the user to interact with the application. For example, a word processing application (such as Microsoft Word) will typically be displayed in a window that has an area for a user to create a document or retrieve and edit a document that is already in existence. If a user wishes to read their email, then they will typically open up an email application (such as Microsoft Outlook) which will typically be displayed in another window that is separate and different from the word processing window. Now, the user has two windows to manage—a word processing window and an email window.

Traditionally, a user has managed multiple windows through the use of a task bar that is, many times, located at the bottom portion of a user display. The task bar is a very thin bar that can extend across a good portion of the user's display and includes references to applications or documents that the user is using or has used. The user can "minimize" the email window when, for example, they wish to work inside the word processing application. When the email window is minimized, a reference to the email application is placed in the task bar. If the user receives an email during the course of working in the word processing application, they can restore the email window by clicking on the reference to the email application in their task bar. This restores the email application so that the user can interact with and read their email. In the course of reading an email

message, the message is likely to be presented in yet another window. If the user chooses to respond to the email message, they typically prepare their response in a "reply" window which is yet a fourth window that the user must manage.

The above-described scenario constitutes a simple example of a window management scenario when the user has opened only two applications. Consider the case where a user has multiple applications (e.g. four or more) that they are working in throughout the day. As a specific example, consider the following four exemplary applications that a user might find necessary to use during the course of their computing day: a word processing application, a presentation application (such as Microsoft Powerpoint), a web browser application (such as Microsoft Internet Explorer), and an email application.

What many users typically do is they open all of the applications and then manage each application's separate and different window as they need access to a different application. The application managing part of the screen uses what is referred to as a "windowing environment." Yet, windowing environments are not necessarily intuitive to all potential computer users. User studies have consistently shown that one of the biggest hurdles for new users of a windowing environment is learning to understand a windowing system, where windows can "layer" on top of each other. Consider, for example, Fig. 1 which shows an exemplary user display 10 that includes four exemplary windows 12, 14, 16, 18, respectively dedicated to a word processor application, a presentation application, a web browser application, and an email application. These windows are layered on top of one another which, for a new computer user, can be a difficult concept to understand and manage. For example, some users might not understand how to use a task bar to manage the windows. They might, for example, inadvertently

close an application when they simply intended to minimize it. Additionally, the users might not appreciate or understand how to move the separate windows around on their display. Further, different windows can sometimes be inadvertently clicked by a user. For example, while a user is in a window associated with one application, they may inadvertently click the edge of a window associated with another application, whereupon they find themselves in the middle of a different application. Finally, many users simply do not comprehend that their screen has multiple layers: they only think of the top window as the one that they can interact with, as if the previous ones were lost.

Windowing environments, however, do not just pose challenges to newer computer users: they can sometimes pose challenges to users who are familiar with such environments. Specifically, management of multiple windows can be distracting to computer users, particularly when a user has many different windows that they are attempting to manage. For example, if a user has many different applications that they are executing that are being managed by a task bar at the bottom of the user's display, to switch from one application to another, the user must find the appropriate task bar portion that references an application of interest. The user must then click on the task bar portion to pull up the application. This scenario can be complicated if, for example, the user has multiple documents in one application that have been minimized, e.g. multiple word processing documents or multiple email messages that they might intend to respond to during the course of their day. This complicates the scenario because now the task bar must maintain an entry for not only each of the user's applications, but each of the documents within each application that might have been minimized by the user as well. At this point, accessing the minimized

applications or document is not an easy task, but rather has devolved into a trial and error hunting process. This is not an efficient way for busy users to manage their applications and documents.

Accordingly, this invention arose out of concerns associated with improving the efficiencies with which a computer user interacts with their computer.

SUMMARY

In one embodiment, a single navigable window is provided and can be navigated by a user to and between different functionalities. A functionality is analogous to an application program. The different functionalities enable a user to accomplish different tasks, e.g. word processing tasks, email tasks, calendar tasks and the like. The single navigable window and the functionalities to which it can be navigated are advantageously provided by a single application that, in turn, provides a very high degree of integration between the functionalities.

In the described embodiment, a user is presented with a user interface (UI) that contains both the single window and navigation instrumentalities that allow the user to navigate between the different functionalities. Exemplary navigation instrumentalities include links associated with each functionality that can be clicked on by a user to access a particular functionality. As the user clicks on various links and engages in different activities within the functionalities, a navigation model maintains a "travel log" of where the user has been so that the user can return to various functionalities to complete tasks or initiate new tasks. In one implementation the travel log is implemented in the form of a navigation stack that utilizes a "back and truncate" model. Advantageously, another of the

exemplary navigation instrumentalities comprises browser-like navigation buttons in the form of a navigation bar that includes "back" and "forward" buttons. By clicking on these buttons, a user can move between functionalities and activities within various functionalities. Various novel navigation model manipulations can be performed to ensure that the user's context is logically maintained. These manipulations include adding, removing, and reorganizing entries.

In one advantageous implementation, context-sensitive commands are automatically presented to a user as they navigate to and between the various functionalities. The context-sensitive commands typically change as the user changes functionalities so that the user always has the correct contextual commands to select from in their currently selected functionality.

In one exemplary embodiment, the functionalities that are available pertain to so-called document-centric functionalities such as word processing functionalities, email functionalities, planner functionalities, web browser functionalities, and the like—all provided within the context of a single application. That is, the functionalities in some way implicate documents that the user is able to browse and/or edit.

In the document-centric embodiment, the documents have properties among which is a document mode. Document modes constrain the way that users interact with documents. Two exemplary modes are "browse mode" and "edit mode". The browse mode allows a user to only view a document and not edit its contents, e.g. a Web page. The edit mode allows a user to edit or change a document, e.g. a word processing document. These modes impact management of the navigation model (i.e. navigation stack) and various inventive measures are taken to ensure that any impact user actions have on the navigation model are

accounted for so that the user's navigation activities bear a logical correspondence with their actions.

In the document-centric embodiment, the user can author different types of documents by navigating the single window to a blank document of a particular type, and then authoring the document in an appropriate manner. The authored document can then be automatically published by the system in the appropriate manner (e.g. saved to a particular collection in the event of a free-form document, transmitted to a recipient in the event of an email message etc.). One particularly advantageous feature of the document-centric single navigable window is that it natively understands and can view different types of document collections. By navigating the window to a particular document collection, the user can select individual documents of the corresponding type and operate upon them in some manner.

In one embodiment, an extensible software platform is provided and serves as a basis to incorporate or integrate different "functionalities" into the single application program that provides the single navigable window. That is, both individual functionalities and the main application itself are extensible.

In one particularly advantageous embodiment, access to functionalities in accordance with the single navigable window application is provided via a network such as the Internet. This provides one basis for a subscriber model service in which the functionalities can be packaged and offered for sale to various subscribers.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is an exemplary display offered by a windowing environment in accordance with the prior art.

Fig. 2 is a high level block diagram of an exemplary computing system that is suitable for implementing the described embodiments.

Fig. 3 is an exemplary user interface (UI) in accordance with one described embodiment.

Fig. 4 is an exemplary specific user interface that is displaying one exemplary document-centric functionality in accordance with an example illustrating the described embodiment.

Fig. 5 is an exemplary specific user interface that is displaying one exemplary document-centric functionality in accordance with an example illustrating the described embodiment.

Fig. 6 is a flow diagram that describes steps in a method in accordance with the described embodiment.

Fig. 7 is a block diagram that illustrates an exemplary navigation model in the form of a navigation stack that implements a "back-and-truncate" model.

Fig. 8 is a block diagram that illustrates an exemplary navigation model in the form of a navigation stack, and visually demonstrates one inventive navigation stack manipulation in accordance with the described embodiment.

Fig. 9 is a flow diagram that describes steps in a method in accordance with the described embodiment.

Fig. 10 is a flow diagram that describes steps in a document-centric method in accordance with the described embodiment.

Fig. 11 is a flow diagram that describes steps in a document-centric method in accordance with the described embodiment.

Fig. 12 is a high level block diagram that illustrates concepts in accordance with one embodiment.

Fig. 13 is a high level block diagram that illustrates concepts in accordance with one embodiment.

DETAILED DESCRIPTION

Overview

The embodiments described just below provide improved methods and systems for creating and using information in a computing environment. The inventive methods and systems address, among other things, user issues relating to computing in a multi-window environment as discussed above.

The inventive methods and systems provide a single navigable window that can be used by a user to navigate to and between multiple different functionalities that are provided by a single application program. By "single navigable window" is meant that users can move from one functionality to another, and within functionalities, all within one window, and then can move back and forth between previously-visited functionalities in a session in a prescribed order based on the order they visited those places. The functionalities enable the user to complete different tasks, and the single window enables the user to navigate between functionalities, and hence tasks, in a seamless manner. By having only one window, the user is relieved of the duties of managing multiple windows. By having all of the functionalities presented within a single application, the user is provided with a highly integrated software product that greatly improves the user's

computing experience. Exemplary functionalities are described in the context of document-centric functionalities such as word processing and email functionalities.

The inventive methods and systems make novel use of a navigation model that manages the user's navigation activities to and between the different functionalities. The described navigation model is a navigation stack that conforms generally to a "back-and-truncate" model.

Navigation instrumentalities are provided and enable the user to navigate among the different functionalities. In the described embodiment these navigation instrumentalities include, without limitation, links to each of the different functionalities as well as browser-like navigation buttons.

Context-sensitive command sets can also be provided in a user interface along with the single navigable window. The context sensitive command sets include commands that automatically change as the user's computing context changes, e.g. as the user moves from functionality to functionality.

In one embodiment, the single application is defined as a software platform that is extensible to receive and incorporate different functionalities. The functionalities can be provided as software modules that can be sent over a network such as the Internet. The extensible software platform provides a basis to offer a subscriber or fee-based service where different subscribers can, for a fee, access different functionalities via a network such as the Internet.

Exemplary Computer System

Fig. 2 shows an exemplary computer system that can be used to implement the embodiments described herein. Computer 130 includes one or more

processors or processing units 132, a system memory 134, and a bus 136 that couples various system components including the system memory 134 to processors 132. The bus 136 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory 134 includes read only memory (ROM) 138 and random access memory (RAM) 140. A basic input/output system (BIOS) 142, containing the basic routines that help to transfer information between elements within computer 130, such as during start-up, is stored in ROM 138.

Computer 130 further includes a hard disk drive 144 for reading from and writing to a hard disk (not shown), a magnetic disk drive 146 for reading from and writing to a removable magnetic disk 148, and an optical disk drive 150 for reading from or writing to a removable optical disk 152 such as a CD ROM or other optical media. The hard disk drive 144, magnetic disk drive 146, and optical disk drive 150 are connected to the bus 136 by an SCSI interface 154 or some other appropriate interface. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for computer 130. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 148 and a removable optical disk 152, it should be appreciated by those skilled in the art that other types of computer-readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk 144, magnetic disk 148, optical disk 152, ROM 138, or RAM 140, including an operating system 158, one or more application programs 160, other program modules 162, and program data 164. A user may enter commands and information into computer 130 through input devices such as a keyboard 166 and a pointing device 168. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to the processing unit 132 through an interface 170 that is coupled to the bus 136. A monitor 172 or other type of display device is also connected to the bus 136 via an interface, such as a video adapter 174. In addition to the monitor, personal computers typically include other peripheral output devices (not shown) such as speakers and printers.

Computer 130 commonly operates in a networked environment using logical connections to one or more remote computers, such as a remote computer 176. The remote computer 176 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 130, although only a memory storage device 178 has been illustrated in Fig. 2. The logical connections depicted in Fig. 2 include a local area network (LAN) 180 and a wide area network (WAN) 182. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, computer 130 is connected to the local network 180 through a network interface or adapter 184. When used in a WAN networking environment, computer 130 typically includes a modem 186

or other means for establishing communications over the wide area network 182, such as the Internet. The modem 186, which may be internal or external, is connected to the bus 136 via a serial port interface 156. In a networked environment, program modules depicted relative to the personal computer 130, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Generally, the data processors of computer 130 are programmed by means of instructions stored at different times in the various computer-readable storage media of the computer. Programs and operating systems are typically distributed, for example, on floppy disks or CD-ROMs. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory. The invention described herein includes these and other various types of computer-readable storage media when such media contain instructions or programs for implementing the steps described below in conjunction with a microprocessor or other data processor. The invention also includes the computer itself when programmed according to the methods and techniques described below.

For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

Single Navigable Window

In accordance with one embodiment, software provides a user interface (UI) that presents a user with a single navigable window that can be navigated from functionality to functionality and inside individual functionalities by a user. The user interface enables the user to effectively manage multiple windows, and hence multiple functionalities, by presenting only one window at a time. This is different from the traditional windowing environment because the windows that pertain to the individual functionalities are not layered on one another and do not need separate management. Another noteworthy way that the single navigable window varies from the traditional windowing environment is that the various functionalities are provided by a single application. That is, in the traditional windowing environment, it is very typical for multiple windows to be provided by multiple different applications that are opened by a user, e.g. a word processing application will have one window, an email application will have another window, a web browsing application will have another window, and the like. All of these windows are separate and requirement separate management by the user. In the present case, various functionalities that were once the domain of individual separate applications are now the domain of a single integrated application which provides its own window management scheme. The window management scheme is embodied in the form of a single navigable window that can be navigated by a user from functionality to functionality.

A user, through the use of navigation instrumentalities can navigate between the functionalities and when doing so, the single window ensures that only one of these functionalities is presented to a user at a time. In the described embodiment, one navigation instrumentality is provided in the form of a web

browser-like navigation tool. The choice of a web browser-like navigation tool follows from concerns that navigation instrumentalities be of a type that is readily understood by most individuals familiar with computing environments. Thus, when a user first encounters the inventive navigable single window concept for the first time, they do not have to learn an unfamiliar navigation concept. Another navigation instrumentality includes links to each of the multiple different functionalities. These links can be clicked on by a user and the single navigable window is automatically navigated to the selected functionality.

Fig. 3 shows but one exemplary user interface (UI) 300 in accordance with one described embodiment. It will be appreciated that other UIs could be used to implement the inventive concepts described herein and that the illustrated UI constitutes but one way of doing so. In the illustrated example, UI 300 includes a navigation bar 302, one or more command areas 304, and a display or document area 306 that constitutes the single navigable window.

Navigation bar 302 is located adjacent the top of display area 306 and contains browser-like navigation buttons 308 in the form of a "backward" button, a "forward" button, a "stop" button and the like. The navigation bar can be located anywhere on the UI. Its illustrated placement, however, is similar in appearance to the placement of traditional web browsing navigation features. In addition to the navigation buttons 308, the navigation bar 302 also includes links 310 to the different functionalities that can be accessed by the user. In the illustrated example, links to three exemplary functionalities (i.e. functionality 1, functionality 2, and functionality 3) are shown. These functionalities are typically different functionalities that can enable a user to complete different respective tasks. Examples of different tasks are given below in more detail. These

functionalities are all provided within the context of a single application. To access a particular functionality, a user simply clicks on one of the links and a window that pertains to the selected functionality is immediately presented in the display area 306.

Command areas 304 are located adjacent the top and left side of the display area 306. The command area(s) can, however, be located in any suitable location. The command areas provide commands that are both global in nature and specific to the particular context the user has selected. For example, some commands such as "search" and "help" might be considered as global in nature since they can find use in many contexts. Other commands, such as "text bold" or "forward" are more specific to the particular context that the user has selected. For the "text bold" command, the user's context may likely be a word processing context, while the "forward" command may likely be employed in an email context. The concept of context-sensitive command structures are described in more detail in the U.S. Patent Application entitled "Task Sensitive Methods And Systems For Displaying Command Sets", incorporated by reference above.

Example

As an example of the single navigable window provided by a single application consider Figs. 4 and 5.

In this example, the multiple functionalities 310 that can be navigated by a user include a browser functionality (indicated by the home icon), a mail functionality (indicated by the letter icon), a planner functionality (indicated by the clock icon), a contacts functionality (indicated by the people icon), a documents functionality (indicated by the folder icon), and a links functionality (indicated by

the world icon). These functionalities are so-called "document-centric" functionalities because they all relate in some way to a document that a user interacts with, e.g. a Web page document, an email document, a calendar document, etc.

Fig. 4 shows an example of a display that is rendered in the display area 306 when a user clicks on the link to the browser functionality. By clicking on the link (i.e. the home icon) to the browser functionality, single application program software executing on the user's computer executes to implement a browser functionality. In this example, the browser functionality displays the user's home page in display area 306. Notice also that navigation buttons 308 are provided for navigation between the different selectable functionalities. The command areas 304 contain command sets that include commands that are specific to the context that the user has selected. In this example, the user's context is a browsing context. Accordingly, the leftmost command area contains commands that are specific to the browsing functionality. Such commands include ones that a user would normally expect to find in a web browser. Notice also that the command area 304 adjacent the top of display area 306 also contains commands that are specific to the browsing context, i.e. "Add to Favorites" and an address well in which the user can type a URL of a particular destination web site.

Fig. 5 shows an example of a display that is rendered in the display area 306 when the user clicks on the link to the mail functionality (i.e. the folder icon). By clicking on this link, single application program software executing on the user's computer executes to implement the mail functionality. In this example, the mail functionality displays a user's in box with messages that have been received by the user. Notice that the leftmost command area has been minimized by the

user and that the command area adjacent the top of the display area 306 contains commands that are specific to the user's current context, e.g. "New" for generating a new email message, "Reply" for replying to an email message, "Reply to All" for replying to all recipients of an email message and the like.

Likewise, although not specifically illustrated, the user could have displays for the planner, contacts, documents, and links functionalities presented in the display area 306 by simply clicking on the links to these specific functionalities. The navigation bar 308 provides the user with the ability to navigate through these different functionalities in a browser-like manner.

It is important to note that the above example constitutes but one exemplary way in which multiple different functionalities can be presented to a user within the construct of a navigable structure. It should be understood that the specifically illustrated functionalities (i.e. browser, mail, planner etc.) constitute specific examples of different functionalities that are capable of being incorporated into the single application program that provides the navigable window. Accordingly, other different functionalities can be employed. This aspect is discussed in more detail in the section entitled "Extensible Functionalities" below.

Fig. 6 is a flow diagram that describes steps in a method in accordance with the described embodiment. The illustrated method can be implemented in any suitable hardware, software, firmware, or combination thereof. In the illustrated example, the method is implemented in software.

Step 600 provides a single application program with multiple different functionalities. The functionalities, as pointed out above, are advantageously different so as to enable a user to accomplish different tasks. One specific non-limiting example of different functionalities was given above in the context of

document-centric functionalities that enable a user to make use of browser, mail, planner, contacts, documents, and links functionalities. Step 600 can be implemented by configuring a computing device, such as a user's computer, with the single application program having the multiple different functionalities. As will be seen in various sections below, this step can also be implemented by providing a software platform in the form of a generic single application shell that is extensible and adaptable to receive different extensions or software modules that embody various different functionalities. These different extensions are then presented to the user in the context of the single application having the multiple different functionalities.

These extensions, as described below, can be delivered to the platform in any suitable way and through any suitable delivery mechanism. For example, one way of delivering the various extensions or functionalities is to deliver them via a network such as an Intranet or the Internet. Regardless of the manner in which the single application is provided, step 602 presents a user interface (UI) with a single window and links to the multiple different functionalities. The UI can also advantageously include navigation instrumentalities that enable a user to navigate between the different functionalities in a browser-like manner. Figs. 3-5 give specific examples of an exemplary UI that can be used in accordance with the described embodiment. Step 604 ascertains whether a user has selected a particular link to a functionality or whether the user has used one of the navigation instrumentalities to navigate to a particular functionality. If a user has not done either, the method branches back to step 602. If, on the other hand, a user has selected a particular link or used a navigation tool to navigate to a particular functionality, step 606 presents a functionality-specific display within the single

window. That is, the single navigable window is navigated by the software to the selected functionality. Specific examples of this were given above in connection with Figs. 4 and 5 in which browsing and mail functionalities were respectively displayed within display area 306. In connection with presenting the functionality-specific display in step 606, step 608 can present functionality-specific commands in a command area of the UI. This is advantageously done automatically as a user navigates from functionality to functionality. That is, as a user changes functionalities, command sets that are specific to the user's current context or functionality are automatically displayed in the command area. Step 608 then branches back to step 604 to ascertain whether the user has navigated to another functionality.

Hence, in the above example, a single application comprises multiple functionalities that enable a user to accomplish different tasks. These multiple functionalities are presented in the context of a single window that is navigable by a user between the different functionalities. Advantageously, navigation instrumentalities are provided that are, in some instances, browser-like in appearance (although not necessarily in behavior, as will be discussed below) and allow the user to navigate between the application-provided functionalities in a browser-like manner. Functionality-specific commands can be automatically presented to the user when they navigate to a particular functionality.

Navigation model

In the described embodiment, a navigation model is utilized to manage a user's navigation activities within the single application that provides the multiple different functionalities. Although any suitable navigation model can be used, in

the described embodiment a so-called "back-and-truncate" navigation stack is used. The basic concept of a back-and-truncate model is known and forms the basis for many different web browsers on the market today. Essentially, the back-and-truncate model makes use of a navigation stack that is truncated when the user navigates back *n* times and then forward to a new document.

Consider, for example, Fig. 7 which illustrates how an exemplary navigation stack works in connection with the described embodiment. Essentially, when a user is presented with the single application UI, they can select links from different areas. First, they might select a push button link (e.g. 310 in Fig. 3) from the navigation bar; second, they might use the forward and back buttons 308; third, they might select a link that is display in the display area 306 (e.g. when in the browsing functionality, there may be links to different web pages that the user can select). As the user makes their way about the various functionalities and selectable links, a navigation stack is built and maintained. In the described embodiment, the navigation stack is maintained in memory, but could easily be maintained in a store to preserve the user's navigation stack for future sessions.

In the Fig. 7 example, a first navigation stack 700 can be established as follows: a user initiates the single window application and is presented with a UI such as the one shown in Fig. 3. From there, a user may click on a link to the first functionality which establishes a first entry 702 in the navigation stack. While engaging the first functionality the user may take part in a certain activity which results in a new display being presented in the display area 306 (Fig. 3). Accordingly, a second entry 704 appears in the navigation stack 700. As the user navigates from functionality to functionality (and takes part in activities within the functionalities) the navigation stack grows and shrinks in accordance with the

exemplary employed back-and-truncate model. This might result in the illustrated entries 706, 708, 710. Assume now that the user employs the "back" button in the navigation bar to move back through the navigation stack. This action is illustrated by arrow A in Fig. 7. When the user arrives at entry 704 in the navigation stack, they click on a different link that navigates their window to functionality 4 (entry 712 in the navigation stack). At this point, the user has moved back through the navigation stack and the stack has been truncated to include only entries 702, 704, and 712. From functionality 4, the user takes part in an activity that presents a display that results in entry 714 in the navigation stack. The above example essentially illustrates the functionality of a back-and-truncate navigation model. In the inventive systems and methods, improvements on this model are made to ensure that the user's navigation experience is logically consistent with actions that the user can engage in.

As an example, the inventive systems and methods provide for navigation stack operations and manipulations that are not found in the typical contemporary navigation models. Specifically, in the present case, the navigation stack can be manipulated to delete, add, and modify navigation stack entries to ensure that consistency is maintained. Consider the following elementary example. Assume that the single window navigation system is employed in the context of functionalities that include the ones described above in connection with Figs. 4 and 5. Assume also that a user navigates from their home page to the email functionality and clicks the "new" button to author a new email message. At this point, the navigation stack will contain entries that look like those in Fig. 8 at 800, 802, and 804. Assume now that the user clicks the "send" button to send the email message to the recipient and then navigates to their planner functionality as

indicated by entry 806. Now assume that the user navigates backward through the navigation stack using their "back" button. If the navigation stack were not manipulated at all, then when the user navigated back to the new email message 804 (which was sent and is physically and logically no longer present), the user would likely receive an error message because the document corresponding to that navigation stack entry is no longer available. Instead, the inventive systems and methods ensure that consistency is maintained by removing the entry 804 and reorganizing the navigation stack so that the entry preceding entry 806 is now entry 802.

Consider another example with reference to Fig. 8. Assume that a user navigates to their email functionality and reads a message that contains a browser link to a web page. Assume also that they click on the browser link which navigates their single window to the web page. The navigation stack will thus contain entries that look like those in Fig. 8 at 810, 812, and 814. Assume now that the user wishes to send an email message to the sender of their original message, but rather than navigating back through the navigation stack by using the "back" button, they simply click the mail link in their navigation bar and thus add entry 816 to the navigation stack. Assume now, while at the email functionality the user decides to delete the mail message that contained the browser link. When the user deletes the email message corresponding to entry 812, the software checks the navigation stack and removes the appropriate entry corresponding to the deleted message, i.e. entry 812. The software then reorganizes the navigation stack so that entry 810 leads to entry 814 and vice versa.

The above examples illustrate an important characteristic of some of the inventive navigation model manipulation operations that distinguish them from

ordinary back-and-truncate operations. For example, here, manipulation of the navigation model, i.e. removing entries, adding entries and reorganizing entries takes place in response to user actions that are not necessarily navigation activities, i.e. the user moving to or between functionalities. Rather, the user's actions that result in the navigation stack manipulations in these examples constitute actions that are supported inherently by the various functionality to which the user has happened to navigate. These user actions necessarily impact the logical association of entries in the navigation model. Without the inventive navigation model manipulation operations, there is a high degree of likelihood that the single navigable window interface would present the user with a degraded experience that is logically inconsistent with the user's navigation activities and/or actions within a particular functionality, e.g. by presenting the user with a logically non-existent document when the user navigates backward in the illustrated navigation stack.

Fig. 9 is a flow diagram that describes steps in a navigation model management method in accordance with the described embodiment. The described method can be implemented in any suitable hardware, software, firmware or combination thereof. In the present example, the method is advantageously implemented in software in connection with a single application that contains multiple functionalities that can be navigated using a single navigable window as described above.

Step 900 establishes a navigation model responsive to user navigation activities. In the example given above, the navigation model is established in memory and manages the user's navigation activities as they navigate from functionality to functionality in the context of a single navigable window. In the

illustrated and described embodiment, the navigation model comprises a navigation stack. Other navigation models can, of course, be used. In step 902, a user takes an action relative to one or more of the functionalities. Here, as was illustrated above, a user's actions can affect different entries in the navigation model depending on the nature of the actions. These actions can affect entries that are displaced many entries away from the user's current location. When these actions impact an entry in the navigation model, the navigation model should be adjusted or otherwise manipulated to ensure that the navigation model always provides the user with a consistent and logically accurate user experience. Accordingly, step 904 ascertains whether a user action has an impact on a navigation model entry. If no entry is impacted by the user's action, then the method returns to step 902 and continues to the next user action. If, on the other hand, step 904 ascertains that the user's action impacts a navigation model entry, then step 906 manipulates one or more of the entries responsive to the user action. Note that this manipulation could be done either by the actions themselves (as shown here, where the actions that manipulate the navigation stack are aware that they need to do so) or by a continuous or conditional monitoring service on the navigation stack. The manipulation of the navigation model can be any manipulation that is directed to maintaining the navigation model's contextual consistency with the user's activities. For example, entries can be removed, added, reorganized, and/or redirected. Further examples of exemplary navigation stack manipulations are given below.

Document centric Application

In the embodiment described in connection with Figs. 4 and 5, the functionalities that are available to the user relate to "document-centric" functionalities. Document-centric functionalities include such things as word processing functionalities, email functionalities, planner functionalities and the like—where the underlying object that the user is operating on is a document. Other document-centric functionalities can, of course, be provided and are not to be limited to those shown in the above examples. Such functionalities can include, without limitation, financial management functionalities, travel functionalities, medical functionalities, income tax functionalities, and telephone log functionalities to name just a few. These document-centric functionalities can be defined by the needs of the user and the creativity of third party software designers that can design extensions for the single window application, as will be explored in the section entitled "Extensible Functionalities" below. In the illustrated example, all of the documents are defined or created in HTML, although other mechanisms of defining or creating document can of course be used.

Document Modes

The document-centric functionalities are characterized, in part in the present example, by the different types of documents that can be encountered. For example, there are "free-form"-type documents such as a word processing document, "form"-type documents such as a "personal contact" form in the contacts functionality where the user can insert certain information in certain blocks, and documents that fall somewhere in the middle between free-form and

form documents, e.g. an email message where there is a form at the top for the addressee and subject, and an open area underneath for text. In the described embodiment, multiple document modes are provided and essentially constrain the actions that a user can take relative to particular documents. Specifically, in the present example there are two document modes that a document can assume—a browse mode and an edit mode.

The document modes are useful because command sets that are displayed in the command area(s) 304 (Fig. 3) may change drastically depending on whether the user is simply viewing a browse-mode document or authoring an edit-mode document. In the present example, the document that is visible in the display area 306 is always in a particular mode. For example, when a document is in browse mode, the user cannot change the underlying HTML that makes up the document. As an example, consider a case where a user navigates to a web page. The web page is displayed in the display area in a browse mode so that the user cannot change the underlying HTML that makes up the web page. (The page may have text areas and other things that can take user input, though.) When, however, a document is in edit mode, the user can change the content (i.e. HTML) that makes up the document. The amount that can be changed depends on the type of document provided: free-form documents allow more parts of the HTML to be edited than form items, for example. Consider that a user might navigate to their word processing functionality to a document collection and pull up a document. This document might be pulled up in browse mode so that the user can read it. If, however, the user wishes to manipulate the document, they can convert it to edit mode to manipulate the content of the document. To convert a document to edit mode, the user need simply click an "edit" button which then enables the user to

edit the document. There is policy that can be defined for when a document should be in edit or browse mode. For example, if the author places a document in edit mode during a session and then navigates to another document without finishing editing activities, then the edit-mode document will be placed in edit mode the next time the author visits the document. In the described embodiment, "new" documents that are created by the user are in edit mode by default. This allows the user to modify the document. In addition, the user can change the mode of a document from edit to browse, e.g. by signifying completion of the document. For example, for a document in the form of an email message, the mode change from edit to browse can be affected by sending the email message. For a document in the form of a contact, the mode change from edit to browse can be affected by saving the contact.

There are some navigation model issues that arise from having different document modes. Consider, for example, a user that navigates the single window to a document that is in browse mode and then converts the document to edit mode. Assume now that the user begins to edit the document but before they are through editing the document, they receive an email message from a friend. Using the navigation function, they click on the mail link to pull up their email box and then click on the new message. At this point, the navigation stack looks like this:

Edit mode document→email box→new message

Assume that when they are finished reading the new message, they wish to return to the document that they were editing. Rather than navigating back through the navigation stack using the "back" button, the user clicks on the

"documents" link to pull up the document functionality and then they click on the individual document that they were editing. Because the document of interest was previously in edit mode (as indicated by the navigation stack), when the document is presented to the user in the display area, it is presented in edit mode and not browse mode. Thus, the system is able to maintain the state of the documents that correspond to navigation model entries even when the user navigates to the documents outside of the direct path defined by the navigation stack.

As another navigation model issue, consider the following: Assume that a user navigates to a document using the single window. The document is located on the Web and is specified by a URL "http://..../document1". Assume now that the user converts the document from browse mode to edit mode. When the user does this, the system makes a copy of the document and places it locally on the user's computer. The local copy of the document is now specified by a URL that is different from the URL that enabled the user to access the document on the Web. The navigation stack is accordingly manipulated to modify the Web-based document URL so that the URL is now the local computer URL. Now if the user moves on and navigates the single window to other functionalities, when they return to document1 by going back through the navigation stack, the navigation stack points to the location of the copy that is currently being edited (i.e. the copy that corresponds to the changed URL) and not the Web-based version of the document. In this instance, the navigation stack is manipulated to point to a different location based upon the user's action. Specifically, the navigation stack is manipulated to modify the URL of an associated entry so that the user's context is consistent when they return to the document associated with that entry.

Authoring

In the document-centric example, authors or users can both create new items and edit many existing items. In the illustrated example, the documents are defined in terms of HTML, although other formats can be used to define the documents. Authoring can take place in a freeform or form format. To author a new item, the user simply clicks a "new" button in the command area 304 (Fig. 3). The user can create items of different types from the same control (i.e. new email types, document types, contact types etc.). For example, a "New" button can provide a dropdown menu to allow the user to choose documents of different types. When a user creates a new document type, the navigation window is navigated to a new empty document of the corresponding type and the new item is entered into the navigation stack in edit mode by default. This new document remains in the navigation stack although provision can be made for maintaining links to the document even if it falls out of the navigation stack.

Fig. 10 is a flow diagram that describes steps in a document creation method in accordance with the described embodiment. The method is advantageously implemented in software. Step 1000 receives user input to author a new document type. The document type can be any suitable document type with exemplary document types including, without limitation, documents, email messages, calendar appointments, contacts and the like. Responsive to the user's input, step 1002 navigates the single window to a new empty document of a corresponding type. Thus, if the user specified a free-form document as the document type, the single window would be navigated to a free-form document; if the user specified an email message as the document type, the single window would be navigated to an empty email message form to be filled in by the user.

Step 1004 then makes an entry in the navigation model (e.g. the navigation stack) corresponding to the new document type. By default, the new document type is entered in the navigation stack in the edit mode so that the user can author the document.

Authors can also edit existing items, as indicated above. Typically, these items are items that they have worked with before. To edit an existing item, the user navigates their single window to the appropriate document and clicks on the "edit" button to convert the document from browse mode to edit mode. In the described embodiment, the navigation stack is manipulated as follows to accommodate the user's action. When the user navigates to a document of interest, an entry is made in the navigation stack that corresponds to the document in browse mode. After all, it was a browse mode document that the user pulled up. When the user clicks the "edit" button, the browse mode document is converted to an edit mode document in the navigation stack. One way of implementing this action is to perform a quick forward and delete operation. That is, when the document is converted into the edit mode, an edit mode entry for a copy of the browse mode document is made in the navigation stack and then the corresponding navigation stack entry for the browse document is deleted. Accordingly, this is but one more example of how the navigation model can be manipulated in accordance with the user's actions to preserve a consistent user context.

Publishing

When a user has completed a document, they can mark the document as complete by publishing it. Publication of a document means different things for

different types of documents. For example, a free-form document might be published to a central server; a contact might be published to the user's own section of the server; an email message is published when it is sent to the recipient. In the described embodiment, the functionalities in the single navigable window application are programmed so that they make sure the single navigable window understands each of the document types with which it is associated and how to specifically publish each of the documents. Thus a user, when finished working on a particular document, might click a "publish" button whereupon the software automatically knows how to publish the corresponding document. This is just one example of how an item might be published. Accordingly, the user need not know protocols or particular locations where the items are to be published.

Fig. 11 is a flow diagram that describes a publishing method in accordance with the described embodiment. The described method is advantageously implemented in software. Step 1100 creates or edits a document. This step is implemented by a user interacting with the single navigable window to either create a new document or edit an existing document as described above. Step 1102 receives user input indicating that the user has completed the document. In the described example, this step is implemented responsive to a user clicking on a button such as a "publish" button. Responsive to the user providing this input, step 1104 automatically publishes the document based upon the document type. In this step, the software is programmed so that the publishing routine that is selected is specifically tied to the document type so that a user need not be concerned with any specific protocols or document locations. Rather, the software automatically publishes the document in a manner that is consistent with the document type.

Viewing Collections

In the described embodiment, one of the advantageous features of the single navigable window application is that it can natively create (via the different built in functionalities), understand, and enable viewing of collections of documents of different types. For example, the application can allow the user to view any collection of documents that the user or others might have created based upon various properties of the documents such as document type or document author. To do this, a user simply navigates the single window to a collection of interest (such as the email box of Fig. 5, or a day on the calendar). The collection then appears in the display area 306 (Fig. 3) of the UI. Once the user has navigated to the particular collection, the software allows, in many cases, selection of a particular item (i.e. selection of a particular mail message in the email box) and navigation of the single window to that item. Thus, the selected item is viewed within the same window as was the particular collection. While at the various collections, the user can be presented with various context-sensitive tools to work within that collection. Examples of context-sensitive tools are described in the U.S. Patent Application entitled "Task Sensitive Methods And Systems For Displaying Command Sets" incorporated by reference above.

Accordingly, in this embodiment, the single navigable window application is able to navigate to and permit viewing of collections of different types of documents. This constitutes a noteworthy departure from conventional browser functionality which does not typically allow navigation and/or user interaction with different types of documents.

Extensible Functionalities

In one embodiment, the functionalities that are provided by the single navigable window application are extensible meaning that the functionalities can be added and removed from the application. In this way, third party developers can provide various extensible functions that can be added or integrated into the single application. In this way, a user can add to their existing set of functionalities by simply selecting one or more functionalities of interest and incorporating the functionalities into the application. In addition, system administrators can add to the existing set of functionalities for a group of users by using a similar selection mechanism. These particular extensible functionalities are referred to as "extensions". Exemplary extensions and methods of providing extensions are described in U.S. Patent Applications entitled "Network-based Software Extensions", and "Architectures For And Methods Of Providing Network-based Software Extensions" incorporated by reference above.

Generic Single Navigable Window Application Platform

One of the advantages of having the extensible functionalities as described above is that now, the single navigable window application can be provided as essentially a shell or software platform upon which these various functionalities can build. Consider for example Fig. 12. There, a single navigable window platform 1200 is shown. Platform 1200, in its most basic form, may have no functionalities whatsoever associated with it. Rather, it may only provide the infrastructure support that implements the single window navigation operability, e.g. software code that provides and manages a navigation model and navigation operations (such as clicking on a link to navigate a window to a particular

functionality). In this case, some or all of the functionalities could then be defined by third parties and added to the platform as appropriate. As an example, functionalities 1202, 1204, and 1206 have been added to the platform to impart a degree of functionality to the single navigable window application. These added functionalities might include a word processing functionality, an email functionality, and a calendar functionality. Other functionalities such as functionalities 1208, 1210, and 1212 have not been but could be added.

Delivery of the functionalities or extensions as they have been referred to above, can be in any suitable way. For example, an individual user might physically insert a CD carrying software code for a particular functionality into their computer and load the software. The functionality would then be added to the platform for the user to interact with. Alternately, the functionalities or extensions can be delivered over a network such as the Internet. An example of how this can be done is described in the Application entitled "Network-based Software Extensions", incorporated by reference above.

Subscription model

In accordance with one embodiment, the single navigable window application and its various functionalities are packaged and provided to consumers as a service to which they can subscribe for a fee. As an example model, consider Fig. 13. There, multiple clients are shown each of which contains a platform such as the one described in connection with Fig. 12. One or more servers are configured to provide various functionalities or extensions for delivery over the Web. The various functionalities or extensions can be provided by independent software vendors and can be written so that they plug directly into the software

platform that provides the single navigable window functionality. Mechanisms for plugging software extensions into a software platform will be understood by those of skill in the art and are not described in detail here.

As an example, consider the case of a small company that employs a computing system to track inventory and perform various office tasks such as word processing, email, accounting (i.e. payroll) and the like. Today, this company might employ several different software applications that provide all of the functionality that is needed for the organization. With that approach, the information technology costs can be very large, particularly for a small company. In the present model, however, a Web-based service ensures that each customer, e.g. each company, can access the various functionalities they need for a subscription fee. The functionalities are plugged into and integrated into a single navigable window application that greatly facilitates the customer's computing experience. It will be appreciated that, although each illustrated client in Fig. 13 is shown as having a platform to which the functionalities can be added, individual platforms for particular customers might be hosted by a separate server so that a customer need only log into the Web to have all of the subscribed functionalities exposed to them.

Conclusion

The embodiments described above provide improved methods and systems for creating and using information in a computing environment. The inventive methods and systems address and provide solutions to user issues relating to computing in a multi-window environment. The described methods and systems provide a single navigable window that can be used by a user to navigate to and

between multiple different functionalities that are provided by a single application program. The functionalities enable the user to complete different tasks, and the single window enables the user to navigate between functionalities, and hence tasks, in a seamless manner. By having only one window, the user is relieved of the duties of managing multiple windows. By having all of the functionalities presented within a single application, the user is provided with a highly integrated software product that greatly improves the user's computing experience.

The described novel use of a navigation model that manages the user's navigation activities to and between the different functionalities ensures that the user's navigation experience bears an accurate logical relationship with the user's various activities. The provided navigation instrumentalities enable the user to navigate among the different functionalities in a quick and efficient manner.

In the software platform embodiment, the single application is extensible to receive and incorporate different functionalities that are provided as software modules that can be sent over a network such as the Internet. The extensible software platform provides a basis to offer a subscriber or fee-based service where different subscribers can, for a fee, access different functionalities via a network such as the Internet.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.